A C T A U N I V E R S I T A T I S L O D Z I E N S I S
FOLIA OECONOMICA 32, 1984

Jerzy Czarnecki*

TECHNIKI PROGRAMOWANIA STOSOWANE W IBM

(na podstawie wewnętrznych publikacji IBM)

Programowanie w znaczeniu programowania operacji komputerowych liczy sobie zaledwie około 30 lat. Owa sztuka lub nauka rozwijała się wraz z rozwojem i zastosowaniem komputerów - od prostych symbolicznych języków programowania do bardzo skomplikowanych. Jak na stosunkowo krótki okres swego istnienia, ta dziedzina ludzkiej działalności charakteryzuje się dużą liczbą błędów i nieporozumień.

Przez długi okres za dobrego programistę uważano kogoś, kto był w stanie napisać program lub jego część w mniejszej liczbie instrukcji niż inni. Do dzisiaj w wielu ośrodkach powstają programy krótkie i "chytre", skomplikowane w swojej wewnętrznej strukturze. Ten sposób pisania miał swoje uzasadnienie w ograniczonej ilości dostępnej pamięci oraz w fakcie, że liczba operacji wykonywana przez maszynę w określonej jednostce czasu była stosunkowo mała. Uzasadnienie to było jednak dobre niegdyś, tymczasem do dzisiaj niektóre programy są niemożliwe do odczytania przez kogokolwiek poza autorem; prawie niemożliwe jest wprowadzenie do nich jakichkolwiek poprawek. W niektórych przypadkach programista ma duże trudności w odczytaniu swojego własnego programu po pół roku, miesiącu, a czasem nawet tygodniu.

Zależy to oczywiście od złożoności programu, niemniej łatwiej jest niekiedy napisać nowy program, niż usunąć błąd logiczny w

*Dr, adiunkt w Zakładzie Teorii Organizacji, Katedra Organizacji i Kierowania UŁ.

istniejącym. Znacznie większy problem stanowi usuwanie błędów, które ujawniają się w użytkowanych programach przy określonej konfiguracji danych; usuwanie błędów powoduje w dużej liczbie przypadków powstawanie nowych błędów.

Na skutek tego rodzaju przykrych doświadczeń, IBM rozpoczął prace nad stworzeniem sposobów programowania, które ułatwiłyby nie tylko tworzenie, ale i użytkowanie istniejących programów. Wynikiem tych prac, trwających ponad 10 lat, jest zestaw technik nazwanych wspólnym mianem "techniki usprawniania technologii programowania". Techniki te mają na celu:

1) zmniejszenie liczby błędów logicznych w powstających programach;

2) uproszczenie, a zarazem zapewnienie czytelności logiki programu;

3) co za tym idzie - ułatwienie nanoszenia poprawek do istniejących programów;

4) nauczenie młodszej kadry programistów dobrych nawyków programowania.

Techniki odnoszą się zawsze do etapu programowania w procesie systemu informatycznego. Stosowanie ich możliwe jest w zaawansowanej fazie rozwoju projektu systemu, kiedy zostały już sprecyzowane wszystkie założenia systemu i wymagania stawiane programom. Stosuje się je na ogół do dużych programów składających się z kilkudziesięciu tysięcy do miliona instrukcji.

U podstaw wszystkich technik, o których będzie mowa, leżą techniki podstawowe stosowane bezwarunkowo we wszystkich zespołach programistów w IBM:

1) technika programowania strukturalnego oraz

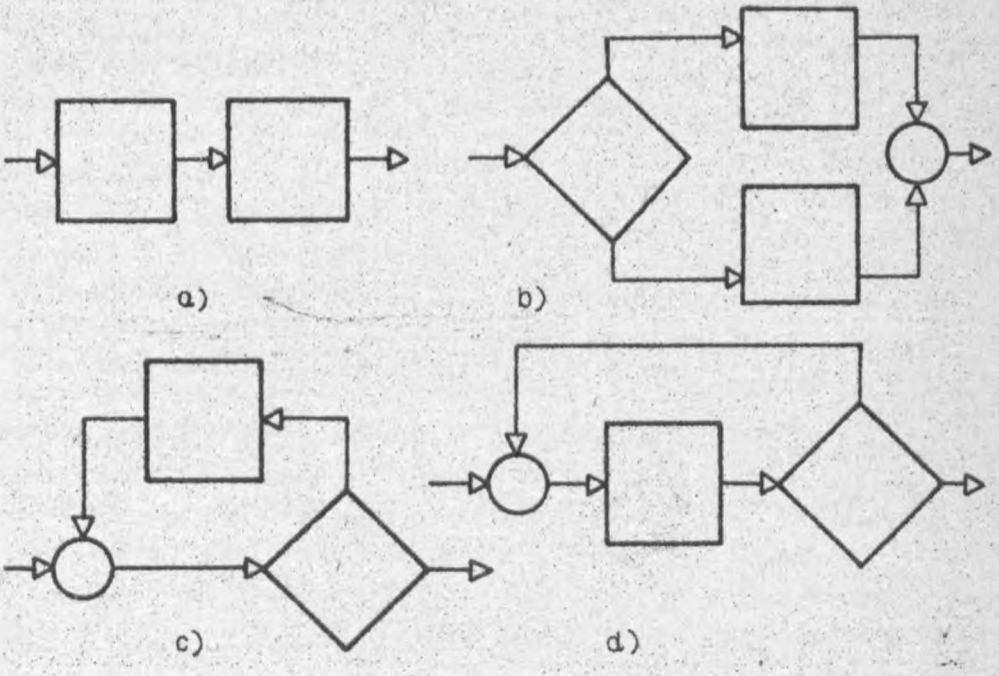
2) technika działania zespołowego.

T e c h n i k a p r o g r a m o w a n i a s t r u k t u r a l n e g o jest główną techniką pisania programów. Powinien ją stosować każdy programista przy konstruowaniu programu. Przedstawia ona pewien sposób myślenia, który można stosować zarówno w fazie projektowania, jak i pisania programu. Zasady programowania strukturalnego umożliwiają stosowanie w procesie programowania innych technik, o których będzie mowa w dalszej części opracowania. Zasady techniki są następujące:

1) w tworzonym programie należy używać tylko kilku struktur logicznych:

- sekwencji jednej lub kilku operacji,
- warunkowego skoku do jednej z dwóch możliwych operacji i powrotu w to samo miejsce (IFTHENELSE),
- powtórzenia operacji przy prawidłowym warunku (DOWHILE lub DOUNTIL).

Struktury te przedstawione są na rys. 1.



Rys. 1. Struktury logiczne:
a) sekwencja; b) IFTHENELSE; c) DOWHILE; d) DOUNTIL

2) program należy podzielić na logiczne segmenty. Każdy z nich wykonuje jedną, logicznie wyodrębnioną funkcję. Każdy segment ma tylko jedno WE i jedno WY;

3) unikać stosowania instrukcji skoków bezwarunkowych (GOTO);

4) segmenty mogą być przywoływane w dowolnym miejscu programu, jednakże główny segment (kontrolujący) powinien przywoływać wyłącznie segmenty poziomu (2), które mogą przywoływać segmenty

poziomu (3) itd. Przywoływanie segmentów odbywa się za pomocą instrukcji CALL, jeśli segment znajduje się wewnątrz programu lub INVOKE, jeśli segment znajduje się w bibliotece systemu;

5) językiem programowania stworzonym specjalnie dla programowania strukturalnego jest PLI, jednakże wszystkie struktury logiczne można symulować w innych językach programowania.

T e c h n i k a d z i a ł a n i a z e s p o ł o w e g o wiąże wszystkie pozostałe techniki w jedną całość poprzez to, że sprawuje w stosunku do nich funkcję kierowniczą. Idea techniki działania zespołowego polega na stworzeniu zespołu programistów "przypisanych" do danego programu we wszystkich fazach jego realizacji. W zespole programistów tworzy się następujące "stanowiska":

1. Główny programista - jest to doświadczony programista i analityk, który ponosi odpowiedzialność za powstawanie programu we wszystkich aspektach. Do jego zadań należy stworzenie głównej funkcji kontrolującej programu i jej własnoręczne zakodowanie; stworzenie i przedstawienie zestawu kodów, które będą stosowane w całym programie; czuwanie nad integracją poszczególnych segmentów powstającego programu.

2. Programista wspomagający - jest to również doświadczony programista i analityk, który wykonuje te same funkcje co główny programista, ale na wyższym stopniu ich szczegółowości (w większym zakresie współpracuje z zespołem na poziomie technicznego wykonywania wszelkich operacji). Programista wspomagający musi być w stanie zastąpić głównego programistę we wszelkich czynnościach w dowolnym momencie.

3. Bibliotekarz - jest to technik lub sekretarka po specjalnym przeszkoleniu. Zadaniem bibliotekarza jest prowadzenie na bieżąco biblioteki systemu. Sporządza on całą dokumentację programu. Jednym z zadań jest zbieranie od programistów zakodowanych programów, kompilacja tych programów oraz ich uruchamianie i testowanie.

4. Programiści - zespół ma charakter płynny, jeśli idzie o liczbę programistów. Każdy z programistów będących członkiem zespołu otrzymuje do wykonania zadanie, za którego wykonanie jest odpowiedzialny. Zadanie to stanowi na ogół segment programu i jest wykonywane na podstawie kodu opracowanego przez głównego programistę.

Główny programista i programista wspomagający są dla wszystkich członków zespołu przełożonymi na czas tworzenia programu i w zakresie zadań wymaganych przy tworzeniu programu. Oznacza to że ani główny programista, ani programista wspomagający nie powinni być przełożonymi służbowymi programistów. Nie mogą też być członkami innego zespołu zadaniowego, podczas gdy programiści mogą być członkami kilku zespołów zadaniowych.

W zespole obowiązuje wysoka dyscyplina wykonywania zadań wyznaczonych przez głównego programistę. Zadania te wymagają na ogół ścisłej współpracy członków zespołu między sobą - rezultaty pracy jednego członka zespołu stanowią podstawę do wykonania zadań przez innego. Dlatego też wszystkie wyniki prac poszczególnych programistów są własnością całego zespołu, a nie pojedynczego programisty, który zabiera ze sobą tajemnice programu odchodząc z ośrodka. Bibliotekarz zespołu nie tylko uruchamia, kompiluje i testuje programy, ale zbiera również wyniki poszczególnych przebiegów, niezależnie od tego, czy rezultaty były dobre czy złe.

Założenia leżące u podstaw wyżej przedstawionych technik umożliwiły stworzenie i zastosowanie w procesie programowania innych technik towarzyszących, które łącznie mają doprowadzić do osiągnięcia wymienionych na wstępie głównych celów. Każda z nich realizuje podobny zestaw celów, niemniej zaleca się stosowanie wszystkich, każdej w innej fazie programowania.

W FAZIE PROJEKTOWANIA, która jest domeną głównego programisty i programisty wspomagającego, stosuje się techniki następujące:

1) technika projektowania strukturalnego oraz
2) technika: Hierarchia-Wejście-Proces-Wyjście (HIPO).

Jakkolwiek obie te techniki są do siebie nieco podobne, to jednak różnią się stopniem szczegółowości.

P r o j e k t o w a n i e s t r u k t u r a l n e ma na celu stworzenie głównych ram programu w sensie naszkicowania funkcji, które program będzie realizował, oraz kolejności realizacji i przywoływania funkcji. Technika stanowi metodykę postępowania przy tworzeniu projektu programu zawierającą się w pięciu logicznych krokach:

1) nakreślenie funkcjonalnego szkicu struktury problemu;
2) określenie zewnętrznych logicznych strumieni danych WE i WY;
3) znalezienie punktów najwyższej abstracji dla danych WE i WY;
4) skonstruowanie podstawowej karty struktury;
5) reiteracja powyższych kroków w stosunku do każdej funkcji niższego poziomu.

HIPO jest techniką pomocniczą przy opisie projektu dla potrzeb zespołu, który musi doskonale rozumieć wszystkie funkcje programu. HIPO jest w pewnym sensie schematem programu z tą różnicą, że schemat blokowy pokazuje organizację i logikę programu, podczas gdy HIPO pokazuje logikę funkcji, które program ma wykonywać. Cała technika sprowadza się do sporządzenia dla programu kilku wykresów:

1) tablicy zawartości,
2) ogólnego diagramu każdej funkcji,
3) szczegółowego diagramu każdej funkcji.

Tablica zawartości niewiele różni się od podstawowej karty struktury będącej wynikiem procesu projektowania programu. Zawiera ona wykres wszystkich funkcji realizowanych przez program wraz z poziomami, z których funkcje te będą przywoływane, legendę stosowanych w niej symboli oraz symboliczny opis zawartości każdego diagramu. Rysunek 2 przedstawia schemat tablicy zawartości.

Ogólny diagram funkcji, podobnie jak szczegółowy diagram funkcji, zawiera opis funkcji. Diagramy pokazują proces, otrzymane rezultaty na wyjściu oraz informacje żądane na wejściu.

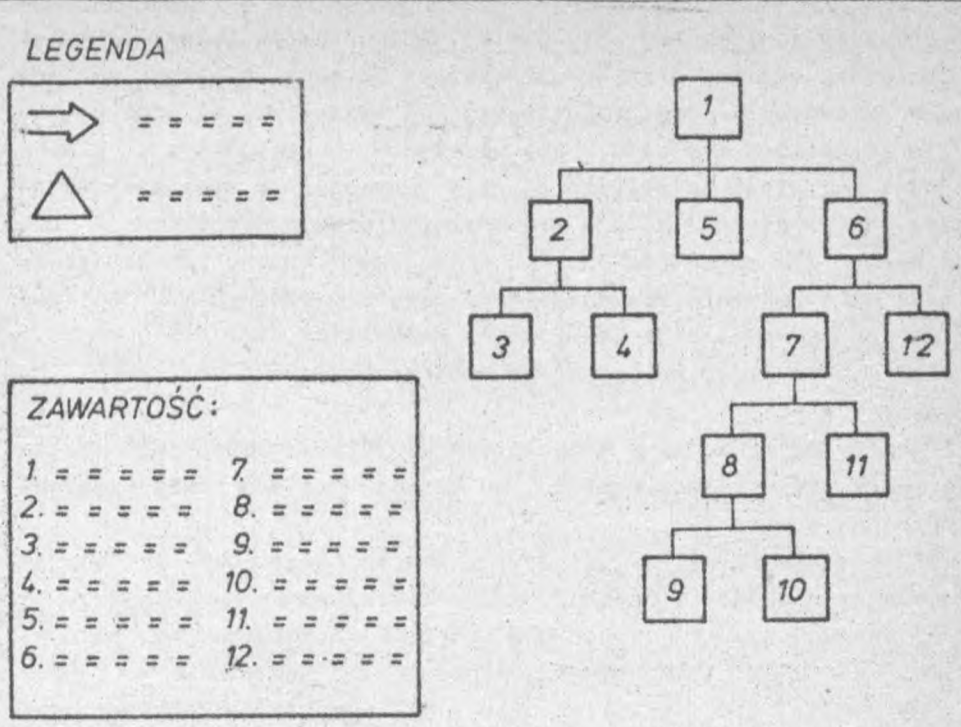
Jakkolwiek technika HIPO tworzona jest i wykorzystywana w fazie projektowania programu, stanowi jednak również jedną z technik dokumentowania programu.

Wyniki obu opisanych wyżej technik mogą okazać się przydatne również i w fazie programowania.

W FAZIE PROGRAMOWANIA, w realizacji której bierze udział cały zespół, stosuje się techniki następujące:

1) technika programowania strukturalnego;
2) technika budowy programu od góry (top-down development);
3) technika przeglądu strukturalnego (structured walk-through).

Jak już wspomniano, technika p r o g r a m o w a n i a s t r u k t u r a l n e g o jest techniką podstawową, dla któ-



Rys. 2. Schemat tablicy zawartości

rej potrzeb skonstruowano inne techniki. Krótki jej opis został przedstawiony na wstępie opracowania i nie będzie w tym opracowaniu rozwijany.

B u d o w a p r o g r a m u o d g ó r y. Na ogół kodowanie (pisanie) programu odbywa się "od dołu". Oznacza to, że każdy segment programu kodowany jest oddzielnie i w taki sam sposób testowany. Na samym końcu, gdy gotowe są wszystkie segmenty, koduje się główną funkcję konstrolującą programu. Ten typ postępowania niesie ze sobą kilka problemów:

1) integracja segmentów niższego szczebla nie jest kompletna;

2) testowanie całości programu może być dokonane dopiero na samym końcu procesu programowania;

3) powoduje to brak możliwości wykrycia ewentualnych błędów powstałych w programie.

Przy zastosowaniu techniki budowy programu "od góry" koduje się w pierwszej kolejności główną funkcję programu. Funkcję tę testuje się następnie, tworząc dla niektórych lub wszystkich funk-

cji niższego poziomu tzw. "korzenie", które symulują funkcje wykonywane przez segment niższego szczebla. Korzenie tworzy się w momencie zakończenia prac projektowych nad segmentem. Na ogół stosuje się je tylko w tym celu, aby sprawdzić, czy funkcja główna wykonuje wszystkie oczekiwane od niej operacje. W momencie testowania funkcji głównej koduje się segmenty drugiego poziomu i buduje korzeń dla segmentów poziomu trzeciego. Proces ten powtarzany jest aż do momentu zakończenia procesu programowania. Opisany sposób postępowania daje następujące korzyści:

1) funkcje programu są integrowane ze sobą już w trakcie ich tworzenia;

2) cały program już w trakcie tworzenia testowany jest kilka razy (segmenty wyższego poziomu po dodaniu każdego nowego segmentu niższego poziomu);

3) znacznie łatwiejszy staje się proces lokalizowania miejsc potencjalnych błędów, ponieważ segmenty testowane są kolejno;

4) czas budowy i testowania programu do momentu jego pełnego uruchomienia ulega znacznemu skróceniu.

P r z e g l ą d s t r u k t u r a l n y. Wszelkie poprawki do programu powinny być naniesione w procesie jego tworzenia, a nie użytkowania. Zostanie to osiągnięte wyłącznie w tym przypadku kiedy wszystkie błędy wykryje programista przed oddaniem programu użytkownikowi. Temu celowi służy technika przeglądu strukturalnego. Stanowi ona serię spotkań zespołu w różnych fazach powstawania programu.

Stosowanie tej techniki opiera się na następujących zasadach:

1) spotkanie jest inicjowane przez twórcę tej części programu lub fazy projektu, która będzie przedmiotem przeglądu;

2) kierownictwo (bezpośredni przełożeni członków zespołu) nie bierze udziału w spotkaniu, gdyż spotkanie nie może stanowić podstawy oceny do pracowników;

3) wszyscy uczestnicy przeglądu otrzymują materiały, które będą przedmiotem przeglądu przed spotkaniem i są zobowiązani do zapoznania się z ich treścią;

4) w trakcie przeglądu kładzie się nacisk na wykrycie błędów, a nie na wskazanie sposobów ich korekty;

5) wszyscy członkowie zespołu - od głównego programisty do programisty - niezależnie od stażu pracy i doświadczenia, przed-

stawiają wyniki swoich prac do przeglądu całemu zespołowi. Od tej zasady nie ma żadnych wyjątków.

W trakcie spotkania uczestnicy przedstawiają swoje uwagi na temat otrzymanego materiału, zgłaszają zastrzeżenia i wątpliwości. W ten sposób ustala się główne obszary dyskusji. Następnie osoba przedstawiająca dokonuje krótkiego opisu przedstawionego materiału, koncentrując się głównie na zgłoszonych uwagach. W końcowej fazie cały zespół pod kierunkiem osoby prezentującej wykonuje funkcje programu "na piechotę". Jeszcze raz dokonuje się przeglądu zadań członków zespołu, z uwzględnieniem aktualnego stopnia zaawansowania prac.

Korzyści płynące z przeglądu są następujące:

1) spotkanie stanowi dobrą motywację dla zespołu. Każdy z jego członków stara się podwyższyć jakość swoich opracowań;

2) spotkanie daje młodszym programistom możliwość zdobycia doświadczenia;

3) wykrywane są wszelkie logiczne błędy projektu i programów.

Zastosowanie techniki przeglądu strukturalnego możliwe jest wyłącznie wtedy, gdy stosuje się technikę działania zespołowego; u podstaw przeglądu leży bowiem założenie, że wszyscy członkowie zespołu rozumieją kod stosowany przez programistę, a nie uczą się go przed każdym spotkaniem w odniesieniu do danego segmentu programu.

W FAZIE DOKUMENTACJI, w realizacji której również bierze udział cały zespół, główny ciężar pracy spoczywa na bibliotekarzu. Technika stosowana w tej fazie powstawania programu nosi nazwę: t e c h n i k a r o z w o j u b i b l i o t e k i w s p o m a g a j ą c e j.

Biblioteka wspomagająca programu składa się z kilku działów:

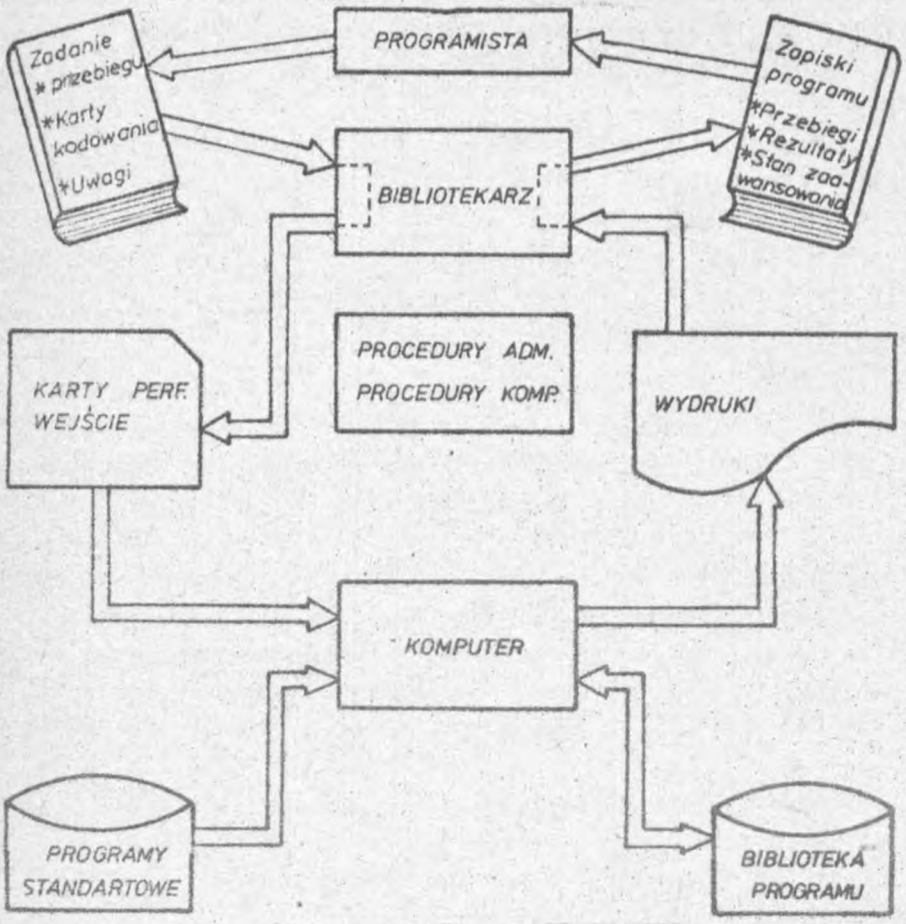
1) zewnętrzna biblioteka programu - w której znajdują się wszelkie opisy analiz programu, projektu programu, wyniki przeglądów itp. - sporządzana jest w formie rękopisów lub maszynopisów;

2) procedury administracyjne i komputerowe. Ta część biblioteki składa się z opisów sposobów realizacji zadań przez bibliotekarza (opisy druków, opisy sposobów uruchamiania i kompilowania programów itp.);

3) wewnętrzna biblioteka programu. Stanowi ona zestaw wszel-

kich składników programu zgromadzonych w pamięci zewnętrznej lub wewnętrznej komputera (programy w postaci skompilowanej, adresy segmentów przywoływanych przez program, które są już zgromadzone w bibliotece wewnętrznej itp.).

Biblioteka programu powstaje we wszystkich fazach jego budowy. Treść jej stanowią karty przebiegu programu, wykresy HIPO, wyniki przeglądów strukturalnych, wyniki pozytywnych testów segmentów programu. W tworzeniu biblioteki uczestniczy cały zespół. Jeśli jest ona prowadzona na bieżąco, stanowi znakomite ułatwienie pracy dla zespołu pracującego nad programem. Umożliwia zapoznanie się z koncepcją programu każdemu spoza zespołu, kto chciałby go uruchomić lub nanieść zmiany. Kluczową rolę w tworzeniu i utrzymaniu biblioteki odgrywa oczywiście bibliotekarz (patrz rys. 3).



Rys. 3. Kluczowa rola bibliotekarza w tworzeniu biblioteki programu

Zestaw technik programowania stosowany w zespołach IBM z pewnością znacznie ułatwia nie tylko proces programowania, ale i proces kształcenia młodszej kadry programistów. Przedstawione techniki mogą sprawiać wrażenie podobnych do siebie, dublujących niektóre sposoby postępowania i osiągane rezultaty. W rzeczywistości każda z nich osiąga inne cele, każda jest przydatna dla któregoś z etapów powstawania programu. Wymagają one zrozumienia przez wszystkich potrzeby i konieczności działania w zespole, zmuszają do działania zorganizowanego; konsekwentnie prowadzą do powstawania programów przejrzystych, czytelnych, łatwych do stosowania i usprawniania.

Bibliografia

Baker F. T., Chief Programmer Team Management of Production Programming, "IBM Systems Journal" 1972, vol. 2, No 1

Baker F. T., System Quality Through Structured Programming, Fall Joint Computer Conference, 1972, vol. 41, part 1

Benson J. P., Structured Programming Techniques, Santa Barbara (Calif.) 1973

Bohm C., Jacopini G., Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules,"Communications of Association for Computing Machinery" 1966, vol. 9, s. 366-371

Dijkstra E. W., Complexity Controlled by Hierarchical Ordering of Function and Variability, Software Engeneering, "NATO Science Commtee Report" January 1969, s. 181-185.

Dijkstra E. W., Notes on Structured Programming, "T. H. Report 70-WSK-03", April 1970

Dijkstra E. W., The Humble Programmer, "Communications of the ACM", October 1972, vol. 15, No 10, s. 859-866.

Elspas B., et al., An Assessment of Techniques for Proving Program Correctness, "Computing Surveys" 1972, No 4

HIPO and Integrated Program Design, "IBM Systems Journal" 1976, vol. 15, No 2 (HIPO - Hierarchical Input Process Output Documentation Technique, IBM, Audio Educational Package)

IBM Instalation Management, An Introduction to Structured Programming in PLI, GC20-1777-0

IBM Structured Programming Workshop, ZR20-4447-0

IBM Structured Programming Workshop Textbook, SR20-7149-1

M i l l s, H., Chief Programmer Teams Principles and Procedures, IBM Federal Systems Div., Gaithersburg (Maryl.) 1971

M i l l s H., Top-Down Programming in Large Systems, Debugging Techniques in Large Systems, "Courant Computer Science Symposium" 1971, 1, NYU s. 41-45

M y e r s G. J., Composite Design: The Design of Modular Programs, Technical Report TR00.2406, IBM Poughkeepsie Laboratory

N e e l y P. M., Fundamentals of Programming Elementary Concepts, University of Kansas, 1973

S t e v e n s V. P., M y e r s G. J., C o n s t a n t i n e L. L., Structured Design, "IBM Systems Journal" 1974, vol. 13, No 2

Jerzy Czarnecki

PROGRAMMING TECHNIQUES EMPLOYED IN IBM

(Based on IBM-materials)

The author distinguishes three basic stages in the programming cycle: designing stage, programming stage, and documentation stage. On the basis of the author's own experience and literature of the subject there are described techniques applied by teams of IBM at particular stages:

1) at designing stage
   - technique of structural designing,
   - HIPO technique;
2) at programming stage
   - technique of structural programming;
   - technique of constructing a programme from top to bottom,
   - technique of structural review;
3) at documentation stage
   - technique of developing an auxiliary library,

with their role and significance for the entire process being stressed. However, a reservation is made that application of these techniques is possible provided a technique of team work is applied at all stages.